Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Λειτουργικά Συστήματα Υπολογιστών
Πρώτη Εργαστηριακή Αναφορά

Ομάδα 25
Λίβα Αναστασία Χριστίνα
Μυστριώτης Γεώργιος

Άσκηση 1

1. Οι επικεφαλίδες προσφέρουν ευελιξία, καθώς καθιστούν διάφορες συναρτήσεις διαθέσιμες στον εκάστοτε προγραμματιστή χωρίς να πρέπει να τις ξαναγράψει. Κάθε συνάρτηση βρίσκεται σε διαφορετικό αρχείο, με το όρισμά της μόνο να περιλαμβάνεται στο header file, συνεπώς είναι πιο εύκολη η μελέτη κάθε μιας συνάρτησης ξεχωριστά. Τέλος, δεν επιβαρύνεται ο μεταγλωττιστής εφόσον σε περίπτωση που αλλαχτεί κάτι σε μια συνάρτηση δεν είναι απαραίτητο να μεταγλωττίσει ξανά όλες τις συναρτήσεις που αποτελούν το πρόγραμμα, αλλά μόνο αυτή που τροποποιήθηκε.

2.
Ακολουθεί το makefile για το εκτελέσιμο του zing

```
all: zing

zing: zing.o main.o
      gcc -o zing zing.o main.o

main.o: main.c
     gcc -Wall -c main.c
```

3.

```
#include <unistd.h>
#include <stdio.h>

char *id;

void zing(){
  id = getlogin();
  printf("Welcome, ");
  while (*id!='\0'){
    printf("%c", *id);
    id++;
  }
printf("\n");
}
```

```
all: zing zing2 fconc

zing: zing.o main.o
        gcc -o zing zing.o main.o

zing2: zing2.o main.o
        gcc -o zing2 zing2.o main.o
```

```
zing2.o: zing2.c
	gcc -Wall -c zing2.c

main.o: main.c
	gcc -Wall -c main.c
```

4. Για να πάρει λιγότερο χρόνο φτιάχνω ένα header file στο οποίο ορίζω όλες τις συναρτήσεις μου, κάθε μία εκ των οποίων είναι υλοποιημένη σε διαφορετικό αρχείο .c. Κάνοντας με ένα makefile compile και link όλα τα αρχεία παίρνω ένα τελικό εκτελέσιμο, στο οποίο αν αλλάξει κάτι σε μια συνάρτηση μεταγλωττίζεται ξανά μόνο η τροποποιημένη συνάρτηση.

5. Με την εντολή
`gcc -Wall -o foo.c foo.c`
δημιουργείται ένα εκτελέσιμο αρχείο με το όνομα “foo.c” οπότε χάνεται οποιοδήποτε άλλο αρχείο με το ίδιο όνομα και γι’ αυτό το λόγο δεν είναι πλέον διαθέσιμος ο πηγαίος κώδικας του εκτελέσιμου.

main:

```
#include "zing.h"
#include <stdio.h>

int main(){
  zing();
  return 0;
}
```

Zing2.c

```
#include <unistd.h>
#include <stdio.h>

char *id;

void zing(){
  id = getlogin();
  printf("Welcome, ");
  while (*id!='\0'){
    printf("%c", *id);
    id++;
  }
printf("\n");
}
```

Η διαδικασία μεταγλώττισης φαίνεται στο makefile που έχει ήδη παρατεθεί.
Λαμβάνουμε τα εξής output:

```
oslaba25@os-node1:~$ ./zing
Hello, oslaba25
```

oslaba25@os-node1:~$ ./zing2
Welcome, oslaba25

Άσκηση 2:

fconc.c:

```
#include "read.h"
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

char *sourcefile1, *sourcefile2, *targetfile;

int main(int argc, char **argv){
        int fd1, fd2;

        char defaultfile[]="fconc.out";

        if(argc<3 || argc>4){
                printf("Usage: ./fconc infile1 infile2 [outfile
(default:fconc.out)]\n");
                return 0;
        }
        else if(argc==3){
                sourcefile1=argv[1];
                sourcefile2=argv[2];
                targetfile=defaultfile;
        }
        else{
                sourcefile1=argv[1];
                sourcefile2=argv[2];
                targetfile=argv[3];
        }
        fd1 = open(sourcefile1, O_RDONLY);
        if (fd1 == -1) {
                perror("open");
                exit(1);
        }
        fd2 = open(sourcefile2, O_RDONLY);
        if (fd2 == -1) {
                perror("open");
                exit(1);
        }
        int fd3, oflags, mode;
        oflags = O_CREAT | O_WRONLY | O_TRUNC;
        mode = S_IRUSR | S_IWUSR;
        fd3 = open(targetfile, oflags, mode);
        if (fd3==-1){
```

```
                perror("open");
                exit(1);
        }


        readtowrite(sourcefile1,targetfile, fd1, fd3);
        readtowrite(sourcefile2,targetfile, fd2, fd3);
        close (fd1);
        close (fd2);
        close (fd3);
        return 0;
}
```

read.h

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

void doWrite (int fdtarget, char *buff, ssize_t len){
       size_t idx;
       ssize_t wcnt;
       idx = 0;

       do { wcnt = write (fdtarget, buff + idx, len-idx);
           if (wcnt==-1){
              perror("write");
              exit(1);
           }
           idx =idx+ wcnt;
        } while (idx < len);

}


void write_file(char *target, char *buff, int fdtarget, ssize_t
rcnt){
      doWrite (fdtarget, buff, rcnt);

      return;
}

void readtowrite(char *readfrom, char *writeto, int fdsource, int
fdtarget){

    char buff[1024];
    ssize_t rcnt;
    for (;;){
        rcnt = read ( fdsource, buff, sizeof(buff)-1);
```

```
        if (rcnt == 0) /* end of file */
            break;

        if(rcnt==-1){ /* error */
            perror("read");
            exit(1);
        }
        //buff[rcnt] = '\0';
        write_file(writeto, buff, fdtarget,rcnt);
    }

    return;
}
```

1. Χρησιμοποιώντας τις εξής εντολές:
oslaba25@os-node1:~$ ./fconc A B C
oslaba25@os-node1:~$ cat C
Hello there!
General Kenobi.
oslaba25@os-node1:~$ strace cat C

Παίρνουμε το αποτέλεσμα
execve("/bin/cat", ["cat", "C"], [/* 26 vars */]) = 0
brk(0) = 0x9e3000
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
mmap(NULL, 8192, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a6a7000
access("/etc/ld.so.preload", R_OK) = -1 ENOENT (No such file or directory)
open("/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=32730, ...}) = 0
mmap(NULL, 32730, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdd2a69f000
close(3) = 0
access("/etc/ld.so.nohwcap", F_OK) = -1 ENOENT (No such file or directory)
open("/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3,
"\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0P\34\2\0\0\0\0\0"...,
832) = 832
fstat(3, {st_mode=S_IFREG|0755, st_size=1738176, ...}) = 0
mmap(NULL, 3844640, PROT_READ|PROT_EXEC,
MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0x7fdd2a0de000
mprotect(0x7fdd2a27f000, 2097152, PROT_NONE) = 0
mmap(0x7fdd2a47f000, 24576, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0x1a1000) = 0x7fdd2a47f000
mmap(0x7fdd2a485000, 14880, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a485000
close(3) = 0
mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a69e000
mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a69d000
mmap(NULL, 4096, PROT_READ|PROT_WRITE,

```
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a69c000
arch_prctl(ARCH_SET_FS, 0x7fdd2a69d700) = 0
mprotect(0x7fdd2a47f000, 16384, PROT_READ) = 0
mprotect(0x60b000, 4096, PROT_READ)      = 0
mprotect(0x7fdd2a6a9000, 4096, PROT_READ) = 0
munmap(0x7fdd2a69f000, 32730)            = 0
brk(0)                                   = 0x9e3000
brk(0xa04000)                                = 0xa04000
open("/usr/lib/locale/locale-archive", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=1859120, ...}) = 0
mmap(NULL, 1859120, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7fdd2a4d6000
close(3)                                 = 0
fstat(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 6), ...}) = 0
open("C", O_RDONLY)                          = 3
fstat(3, {st_mode=S_IFREG|0600, st_size=29, ...}) = 0
fadvise64(3, 0, 0, POSIX_FADV_SEQUENTIAL) = 0
mmap(NULL, 270336, PROT_READ|PROT_WRITE,
MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7fdd2a09c000
read(3, "Hello there!\nGeneral Kenobi.\n", 262144) = 29
write(1, "Hello there!\nGeneral Kenobi.\n", 29Hello there!
General Kenobi.
) = 29
read(3, "", 262144)                          = 0
munmap(0x7fdd2a09c000, 270336)           = 0
close(3)                                 = 0
close(1)                                 = 0
close(2)                                 = 0
exit_group(0)                               = ?
+++ exited with 0 +++
```